This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1-33 (cancelled).

34 (original). A system for producing results from a database comprising:

a spooling module which receives a relation comprising a set of rows and a set of columns, and which copies into a memory location one or more rows of said relation having common values for one or more columns of said relation; and

an application module which invokes a relational procedure, wherein said relational procedure performs a relational operation on rows in said memory location to produce one or more result rows, said application module invoking said spooling module when said relational procedure is unable to produce result rows.

35 (original). The system of claim 34, further comprising:

a query compiler which receives a query and produces an expression tree based on said query, said relational procedure being based on said expression tree.

36 (original). The system of claim 35, wherein said expression tree comprises a join operator having:

a first operand which includes a first instance of a relation, said relation having a set of rows and a set of columns;

a second operand including a second instance of said relation; and

a first predicate;

said system further comprising:

an optimization module which receives said expression tree and determines that said predicate is, or conjunctively includes, an equality comparison between one or more columns of said first instance of said relation and corresponding one or more columns of said

second instance of said relation, said one or more columns of said first and second instances of said relation being the same columns as said one or more columns of said relation.

37 (original). The system of claim 34, wherein said relational procedure comprises a join.

38 (original). The system of claim 37, wherein said join comprises one of: inner join, semijoin, or anti-semijoin.

39 (original). A method of evaluating a semijoin having a first operand, a second operand, and a predicate, said method comprising the acts of:

determining that said first and second operands each comprise first and second instances, respectively, of a common relation;

determining that said predicate is, or conjunctively includes, an equality comparison between one or more columns of said first instance of said relation and corresponding columns of said second instance of said relation;

segmenting said common relation based on said one or more columns to produce one or more segments of said common relation; and

performing said semijoin separately on each of said segments.

40 (original). The method of claim 39, further comprising the act of spooling each of said segments in a memory location, wherein said performing act comprises applying said semijoin successively to the segments spooled in said memory location.

41 (original). The method of claim 39, further comprising the act of compiling a SQL query which includes an EXISTS clause to produce a relational expression that includes said semijoin.

42 (original). A computer-readable medium having computer-executable instructions to perform the method of claim 39.

43 (original). A method of evaluating an anti-semijoin having a first operand, a second operand, and a predicate, said method comprising the acts of:

determining that said first and second operands each comprise first and second instances, respectively, of a common relation;

determining that said predicate is, or conjunctively includes, an equality comparison between one or more columns of said first instance of said relation and corresponding columns of said second instance of said relation;

segmenting said common relation based on said one or more columns to produce one or more segments of said common relation; and

performing said anti-semijoin separately on each of said segments.

44 (original). The method of claim 43, further comprising the act of spooling each of said segments in a memory location, wherein said performing act comprises applying said anti-semijoin successively to the segments spooled in said memory location.

45 (original). The method of claim 43, further comprising the act of compiling a SQL query which includes a NOT EXISTS clause to produce a relational expression that includes said anti-semijoin.

46 (original). A computer-readable medium having computer-executable instructions to perform the method of claim 43.

47 (original). A method of sorting rows of a database table according to a first set of one or more columns, said rows having been sorted on a second set of one or more columns, said method comprising the acts of:

segmenting said database table based on said second set of columns to produce segments of said database table, each of said segments comprising rows having common values in said second set of columns; and

separately sorting each of said segments based on the values in said first set of columns.

48 (original). The method of claim 47, wherein said segmenting act is performed without spooling rows of said segments.

49 (original). The method of claim 47, wherein said segmenting act comprises:

identifying a first row of said database table, said first rows having first values in said second set of columns; and

identifying a second row of said database table, said second row having said first values in said second set of columns, said database table having an order, said second row being the last occurring row in said order having said first values in said second set of columns.

50 (original). The method of claim 47, wherein said act of separately sorting comprises:

invoking a first function which sorts rows in first one of said segments;

invoking a second function which identifies a next one of said segments; and

repeating said acts of invoking said first and second function until all of said segments have been exhausted.

51 (original). The method of claim 47, further comprising the act of:

creating an expression tree including a GbApply operator having:

a first child sub-tree specifying said database table as input to said GbApply operator;

segmentation data indicating that said database table is to be segmented on said second set of columns; and

a second child sub-tree specifying a relational fragment to be performed on each of said segments, said relational fragment specifying a sort of the rows of said segments on said first set of columns.

52 (original). A computer-readable medium having computer-executable instructions to perform the method of claim 47.

53 (original). In a database system which performs operations on a table having rows and columns, a method of identifying a row having a superlative value in a first of said columns from among a set of rows having a common value in a second of said columns, said method comprising the acts of:

segmenting said rows based on the values in said second column to produce groups of one or more rows, wherein all of the rows in a first of said groups have a common value in said second column;

sorting the rows in said first group based on the values in said first column; and

identifying the first or last row in said first group.

54 (original). The method of claim 53, wherein said superlative value comprises a maximum value, and wherein said identifying act comprises identifying the last row in said first group.

55 (original). The method of claim 53, wherein said superlative value comprises a minimum value, and wherein said identifying act comprises identifying the first row in said first group.

56 (original). The method of claim 53, wherein said segmenting act comprises:

sorting said rows based on the values in said second column; and

following said sorting act, identifying the first and last rows which have said common value in said second column.

57 (original). The method of claim 53, further comprising the act of:

creating an expression tree including a GbApply operator having:

a first child sub-tree specifying said table as input to said GbApply

operator;

segmentation data indicating that said database table is to be segmented on said second set of columns; and

a second child sub-tree specifying a relational fragment to be performed on each of said segments, said relational fragment specifying the selection of either the first or last of the rows in a segment.

58 (original). A computer-readable medium having computer-executable instructions to perform the method of claim 53.

59 (original). A system for evaluating a database query comprising:

a query compiler which receives the query and produces an expression tree based on the query;

an analysis module which identifies an expression tree having a join operator which includes:

a first operand which includes a first instance of a relation based on information stored in said database, said relation having a set of rows and a set of columns;

a second operand including a second instance of said relation; and

a first predicate which is, or conjunctively includes, an equality comparison between one or more columns of said first instance of said relation and corresponding one or more columns of said second instance of said relation; and

an evaluation module which segments said relation according to the values in said one or more columns and which applies said join operator separately to each segment of said relation.

60 (original). The system of claim 59, further comprising:

a spooling module which spools rows of each segment in a designated memory location;

wherein said evaluation module applies said join operator to the rows stored in said designated memory location.

61 (original). The system of claim 59, wherein said evaluation module does not spool the segments of said relation.

62 (original). The system of claim 59, wherein said join operator comprises one of: inner join, semijoin, or anti-semijoin.

63 (original). The system of claim 59, wherein said analysis module determines that said first operand further includes a filter which modifies said first instance of said relation according to a second predicate.

64 (original). The system of claim 59, wherein said analysis module determines that said first operand further includes an aggregate operation which specifies the computation of a value based on one or more rows of said relation.

65 (original). The system of claim 59, wherein said relation comprises a stored table.

66 (original). The system of claim 59, wherein said relation comprises a sub-expression based on one or more stored tables, and wherein said system further includes a module which evaluates said sub-expression to produce said relation.